AMENDMENTS TO THE CLAIMS

Please amend the claims as follows:

27. (Currently Amended) A computer-based method for compiling a structured document schema into type annotation records comprising steps of:

a. building a type hierarchy ordered tree from structured document ~~with each type record node in said hierarchy ordered tree containing a typing tuple of the following format: <*string_value*, *type*>~~based on a derivation of relations among types in said structured document and determining one or more tuples for each type record in said structured document;

b. ~~creating~~ forming a complete typing ~~SET which contains all typing tuples in said type hierarchy ordered tree~~set of said tuples;

c. sorting said typing ~~tuples in said SET~~ set by their first field~~, *string_value*~~;

d. creating, from sorted tuples in (c), ambiguity typing sequences for tuples having a common first field~~, *string_value*~~ and having a unique second field, collecting and sorting a third field from ambiguity typing sequences, assigning a unique offset number to each sorted third field, and arranging said ambiguity typing sequences based on offset numbers;

~~**e.** sorting and patching ambiguity typing sequences with dummy items;~~

~~**f.**~~**e.** creating a typing array by concatenating typing tuples in ~~said sorted and patched~~ resulting ambiguity typing sequences of (d);

~~**g.**~~**f.** for each type record node, N, in created typing array, if the intersection of a set of tuples in N with any ambiguity typing sequence is not empty, then replacing first typing tuple ~~<*string_n*, *type_n*>~~ in N by ~~(*string_n*, *type_n*, *offset*>~~typing tuple having offset, wherein offset represents a position of an ambiguity type in a given ambiguity typing sequence;

~~h.~~g. creating ~~an index structure to link each string value to its type~~a type indexing data structure and indicating ambiguity type for each type name; and

~~i.~~h. outputting said created index structure.

28. (Original) The computer-based method of claim 27, wherein said structured document schema is an XML document schema.

29. (Canceled)

30. (Original) The computer-based method of claim 27, wherein said index structure is any of the following: hash table, binary tree, or B+ tree.

31. (Original) The computer-based method of claim 27, wherein said computer-based method is implemented in a database.

32. (**Canceled**).

33. (Currently Amended) An article of manufacture comprising ~~a computer usable medium~~ having computer readable program code embodied therein which implements a method for compiling a structured document schema into type annotation records, said ~~computer usable medium comprising steps of~~computer readable program code comprising:

a. computer readable program code building a type hierarchy ordered tree from structured document ~~with each type record node in said hierarchy ordered tree containing a~~

~~typing tuple of the following format: <*string_value*, *type*>~~ based on a derivation of relations among types in said structured document and determining one or more tuples for each type record in said structured document;

b. computer readable program code ~~creating~~ forming a complete typing ~~SET which contains all typing tuples in said type hierarchy ordered tree~~set of said tuples;

c. computer readable program code sorting said typing ~~tuples in said SET~~ set by their first field~~, *string_value*~~;

d. computer readable program code creating, from sorted tuples in (c), ambiguity typing sequences for tuples having a common first field~~, *string_value*~~ and having a unique second field, collecting and sorting a third field from ambiguity typing sequences, assigning a unique offset number to each sorted third field, and arranging said ambiguity typing sequences based on offset numbers;

~~**e.** computer readable program code sorting and patching ambiguity typing sequences with dummy items;~~

~~**f.**~~**e.** computer readable program code creating a typing array by concatenating typing tuples in ~~said sorted and patched~~ resulting ambiguity typing sequences of (d);

~~**g.**~~**f.** computer readable program code for each type record node, N, in created typing array, if the intersection of a set of tuples in N with any ambiguity typing sequence is not empty, then replacing first typing tuple ~~<*string_n*, *type_n*>~~ in N by ~~(*string_n*, *type_n*, *offset*>~~ typing tuple having offset, wherein offset represents a position of an ambiguity type in a given ambiguity typing sequence;

~~h.~~**g.** computer readable program code creating ~~an index structure to link each string_value to its type~~a type indexing data structure and indicating ambiguity type for each type name; and

~~i.~~**h.** computer readable program code outputting said created index structure.

34. (Original) The article of manufacture of claim 33, wherein said structured document schema is an XML document schema.

35. (Canceled)

36. (Original) The article of manufacture of claim 33, wherein said index structure is any of the following: hash table, binary tree, or B+ tree.

37. (Canceled)

38. (Canceled)

39. (Currently Amended) A computer-based method for compiling a structured document schema into type annotation records comprising steps of:

a. building a type hierarchy ordered tree from XML document schema ~~with each type record node in said hierarchy ordered tree containing a typing tuple of the following format: <*string_value*, *type*>~~ based on a derivation of relations among types in said structured document and determining one or more tuples for each type record in said structured document~~;~~

b. ~~creating~~ forming a complete typing ~~SET which contains all typing tuples in said type hierarchy ordered tree~~ set of said tuples;

c. alphabetical sorting said typing ~~tuples in said SET~~ set by their first field~~,~~ ~~*string_value*~~;

d. creating, from sorted tuples in (c), ambiguity typing sequences for tuples having a common first field~~, *string_value*~~ and having a unique second field, collecting and sorting a third field from ambiguity typing sequences, assigning a unique offset number to each sorted third field, and arranging said ambiguity typing sequences based on offset numbers;

~~**e.** sorting and patching ambiguity typing sequences with dummy items;~~

~~**f.**~~**e.** creating a typing array by concatenating typing tuples in said ~~sorted and patched~~ resulting ambiguity typing sequences of (d);

~~**g.**~~**f.** for each type record node, N, in created typing array, if the intersection of a set of tuples in N with any ambiguity typing sequence is not empty, then replacing first typing tuple ~~*<string_n, type_n>*~~ in N by ~~*(string_n, type_n, offset>*~~ typing tuple having offset, wherein offset represents a position of an ambiguity type in a given ambiguity typing sequence;

~~**h.**~~**g.** creating any of the following ~~index data structures to link each string_value to its type~~type indexing data structures and indicating ambiguity type for each type name: hash table, binary tree, or B+ tree; and

~~**i.**~~**h.** outputting said created index structure.

40. (Canceled)

41. (Original) The computer-based method of claim 39, wherein said computer-based method is implemented in a database.